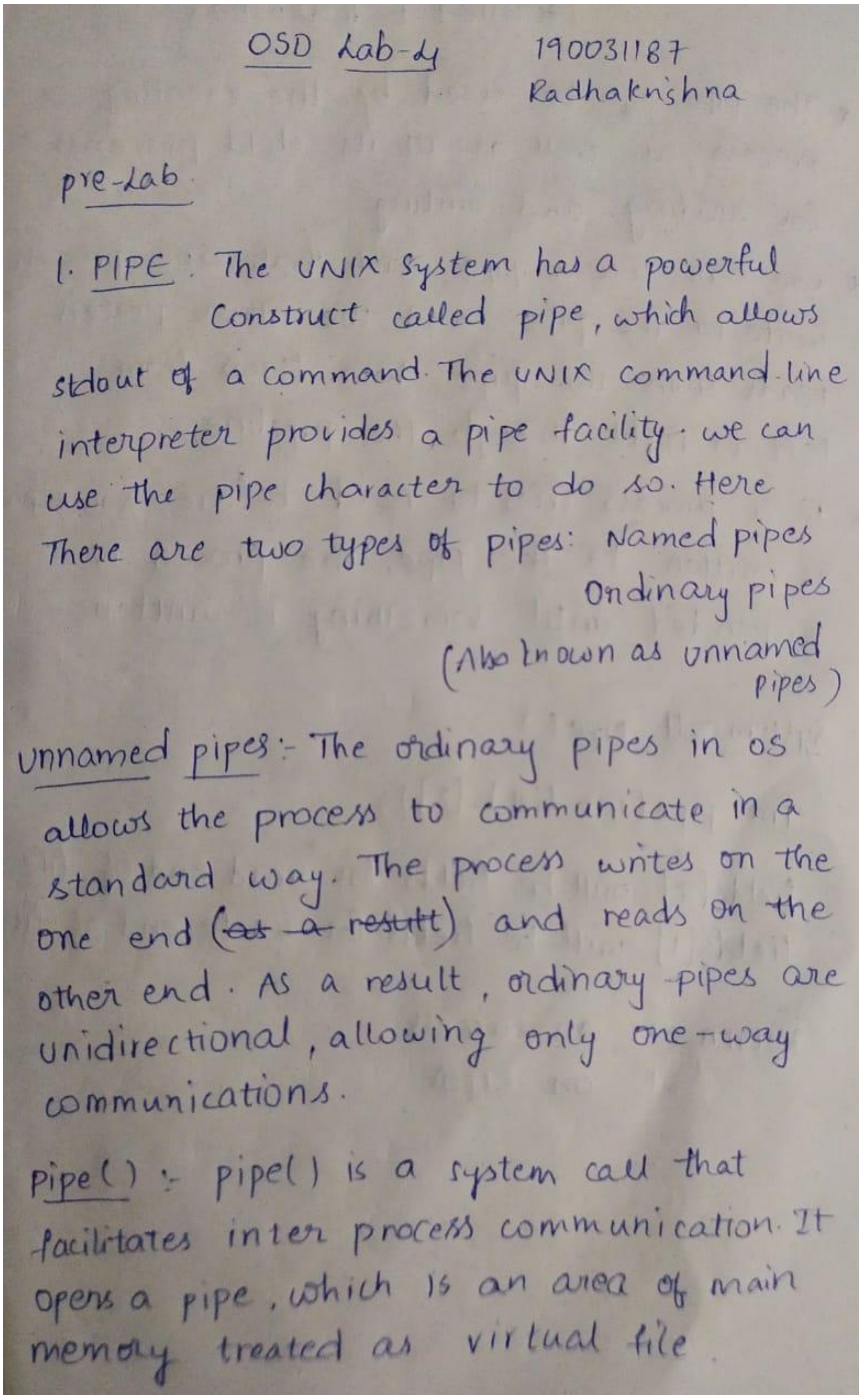
Operating System and Design (19CS2106S)
Lab- 4
Pre-Lab:

Unnamed Pipes: pipe ()



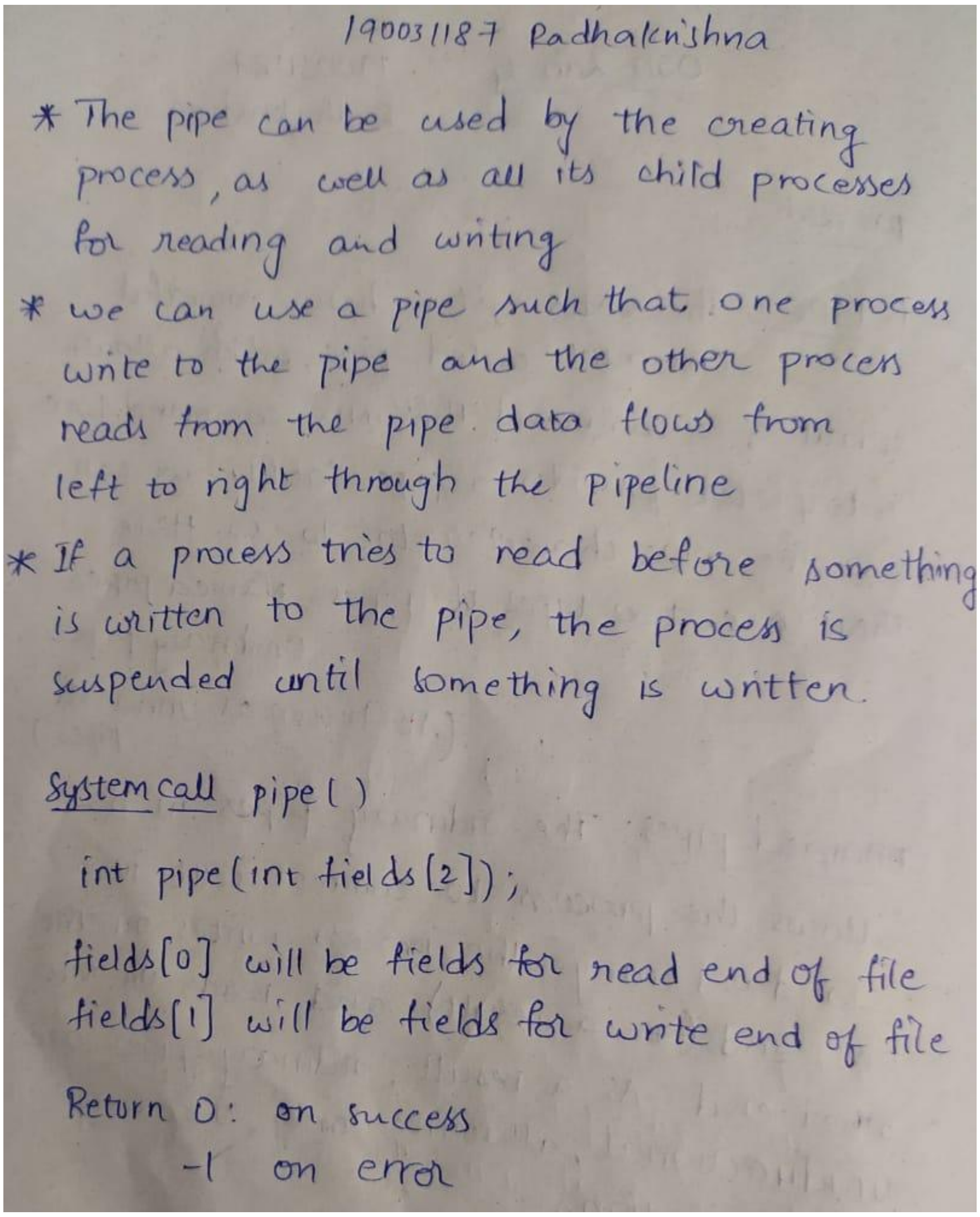
OSD Lab-4 190031187
Radhakrishna

pre-Lab

1. PIPE : The UNIX System has a powerful Construct called pipe, which allows stdout of a command. The UNIX command line interpreter provides a pipe facility. we can use the pipe character to do so. Here There are two types of pipes: Named pipes
Ordinary pipes
(Also known as unnamed pipes)

unnamed pipes:- The ordinary pipes in os allows the process to communicate in a standard way. The process writes on the one end (~~as a result~~) and reads on the other end. As a result, ordinary pipes are unidirectional, allowing only one-way communications.

pipe() :- pipe() is a system call that facilitates inter process communication. It opens a pipe, which is an area of main memory treated as virtual file.



190031187 Radhakrishna

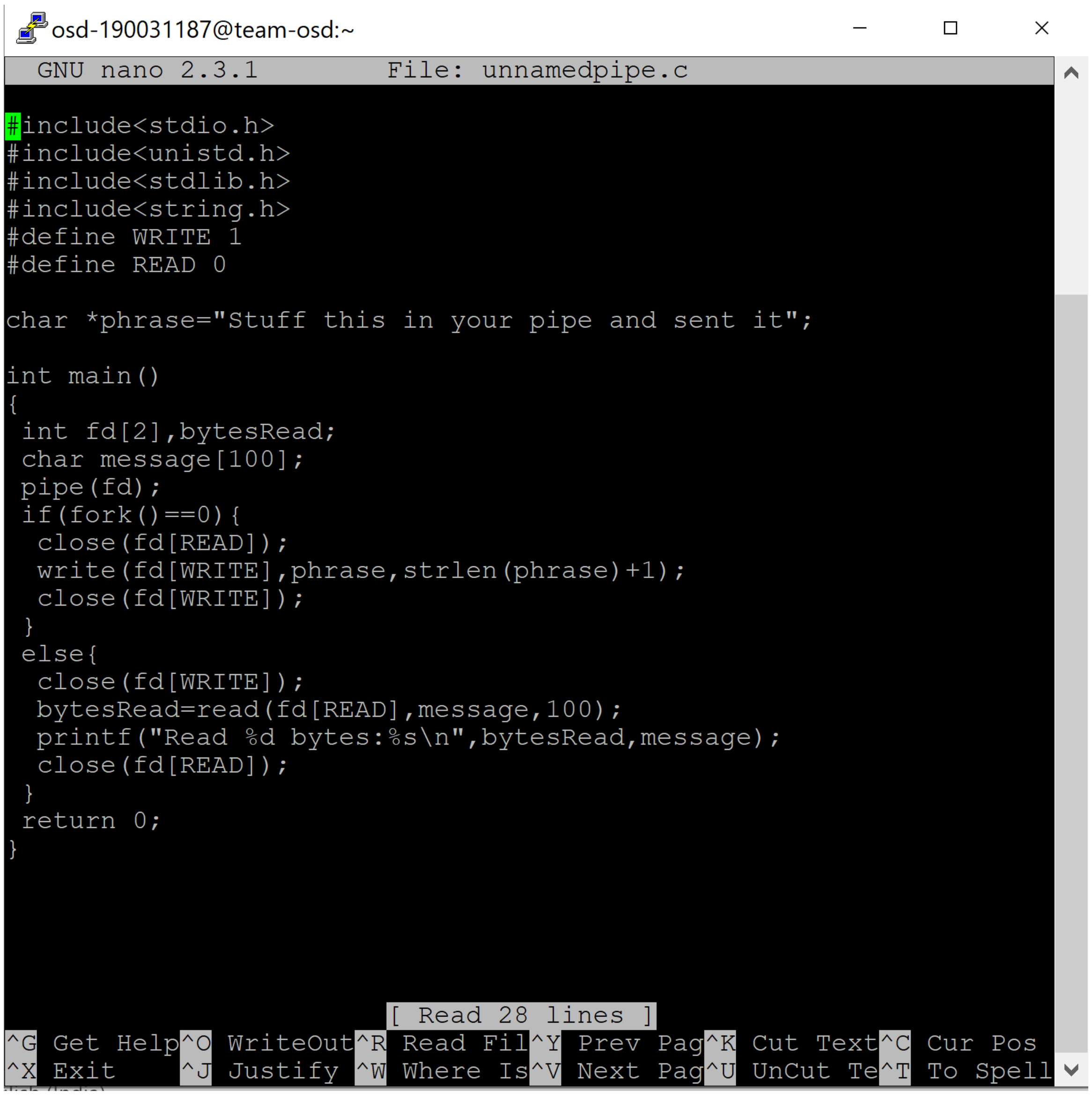
* The pipe can be used by the creating process, as well as all its child processes for reading and writing
* we can use a pipe such that one process write to the pipe and the other process reads from the pipe. data flows from left to right through the pipeline
* If a process tries to read before something is written to the pipe, the process is suspended until something is written.

<u>System call</u> pipe ( )

int pipe(int fields[2]);

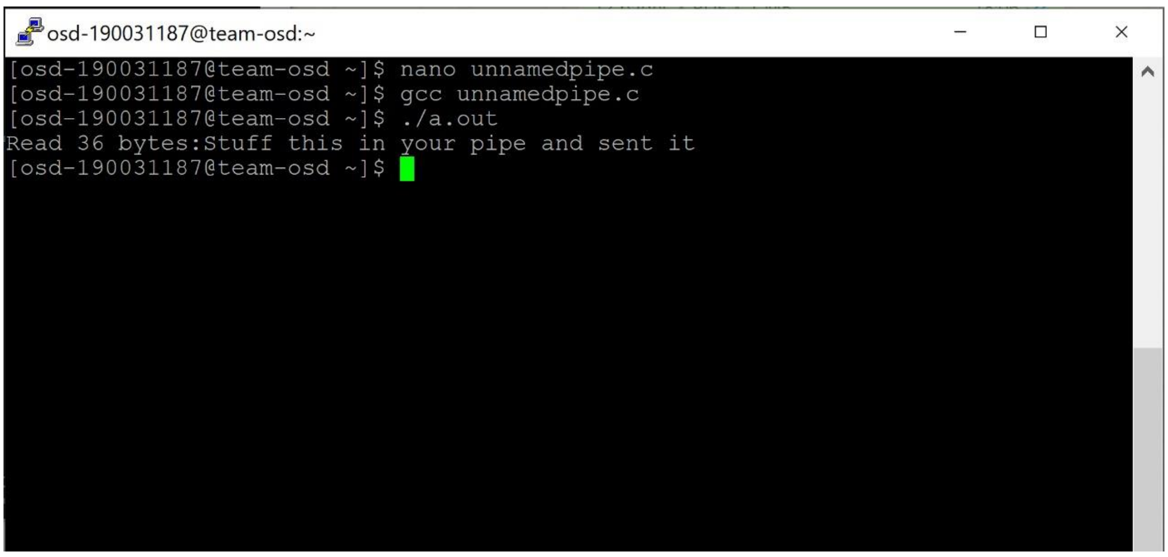
fields[0] will be fields for read end of file
fields[1] will be fields for write end of file

Return 0 : on success
-1 on error



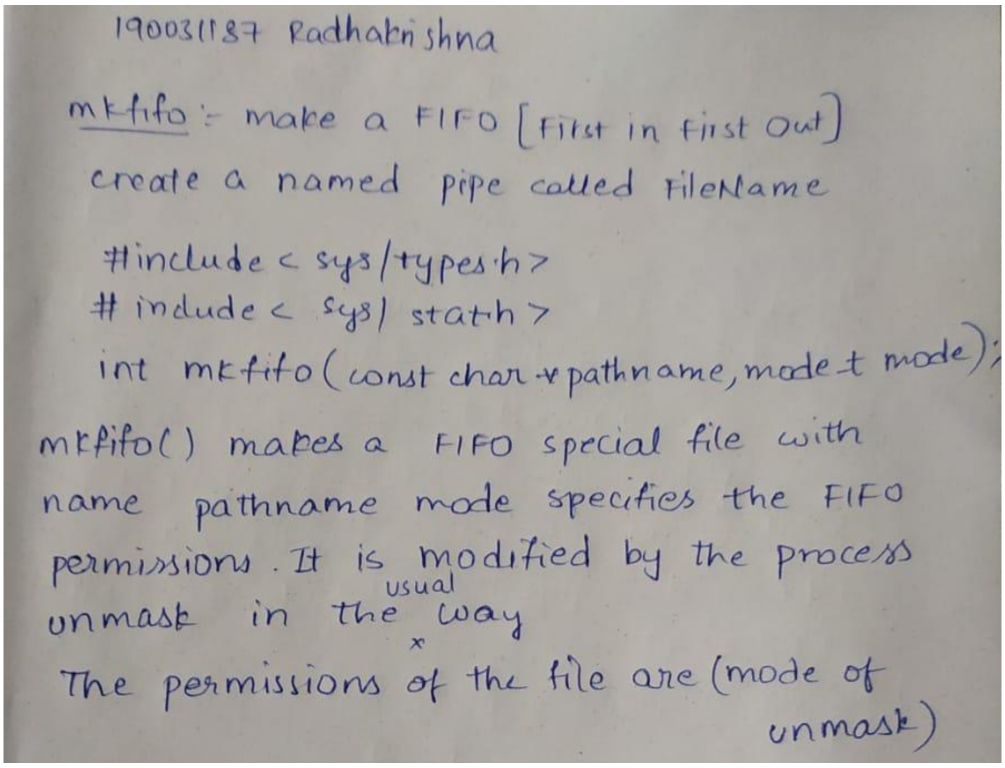
```
  GNU nano 2.3.1          File: unnamedpipe.c

#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
#define WRITE 1
#define READ 0

char *phrase="Stuff this in your pipe and sent it";

int main()
{
 int fd[2],bytesRead;
 char message[100];
 pipe(fd);
 if(fork()==0){
  close(fd[READ]);
  write(fd[WRITE],phrase,strlen(phrase)+1);
  close(fd[WRITE]);
 }
 else{
  close(fd[WRITE]);
  bytesRead=read(fd[READ],message,100);
  printf("Read %d bytes:%s\n",bytesRead,message);
  close(fd[READ]);
 }
 return 0;
}

                                    [ Read 28 lines ]
^G Get Help^O WriteOut^R Read Fil^Y Prev Pag^K Cut Text^C Cur Pos
^X Exit    ^J Justify ^W Where Is^V Next Pag^U UnCut Te^T To Spell
```

OUTPUT



```
[osd-190031187@team-osd ~]$ nano unnamedpipe.c
[osd-190031187@team-osd ~]$ gcc unnamedpipe.c
[osd-190031187@team-osd ~]$ ./a.out
Read 36 bytes:Stuff this in your pipe and sent it
[osd-190031187@team-osd ~]$
```

mkfifo: creates a named pipe called fileName.



190031187 Radhakrishna

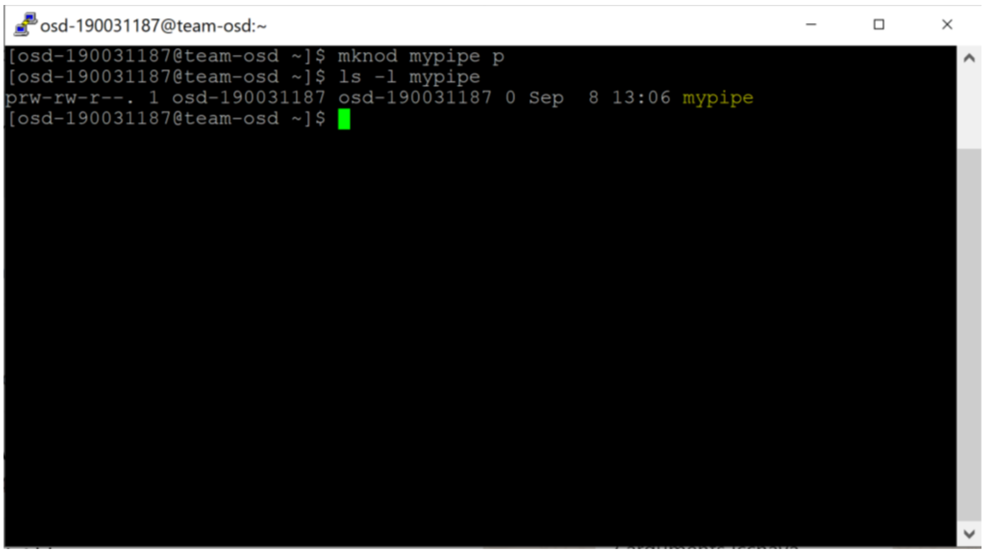
mkfifo :- make a FIFO [First in First Out]
create a named pipe called FileName

```
#include < sys/types.h >
# include < sys/ stat.h >
int mkfifo(const char * pathname, mode_t mode);
```

mkfifo() makes a FIFO special file with name pathname mode specifies the FIFO permissions. It is modified by the process unmask in the usual way

The permissions of the file are (mode of unmask)

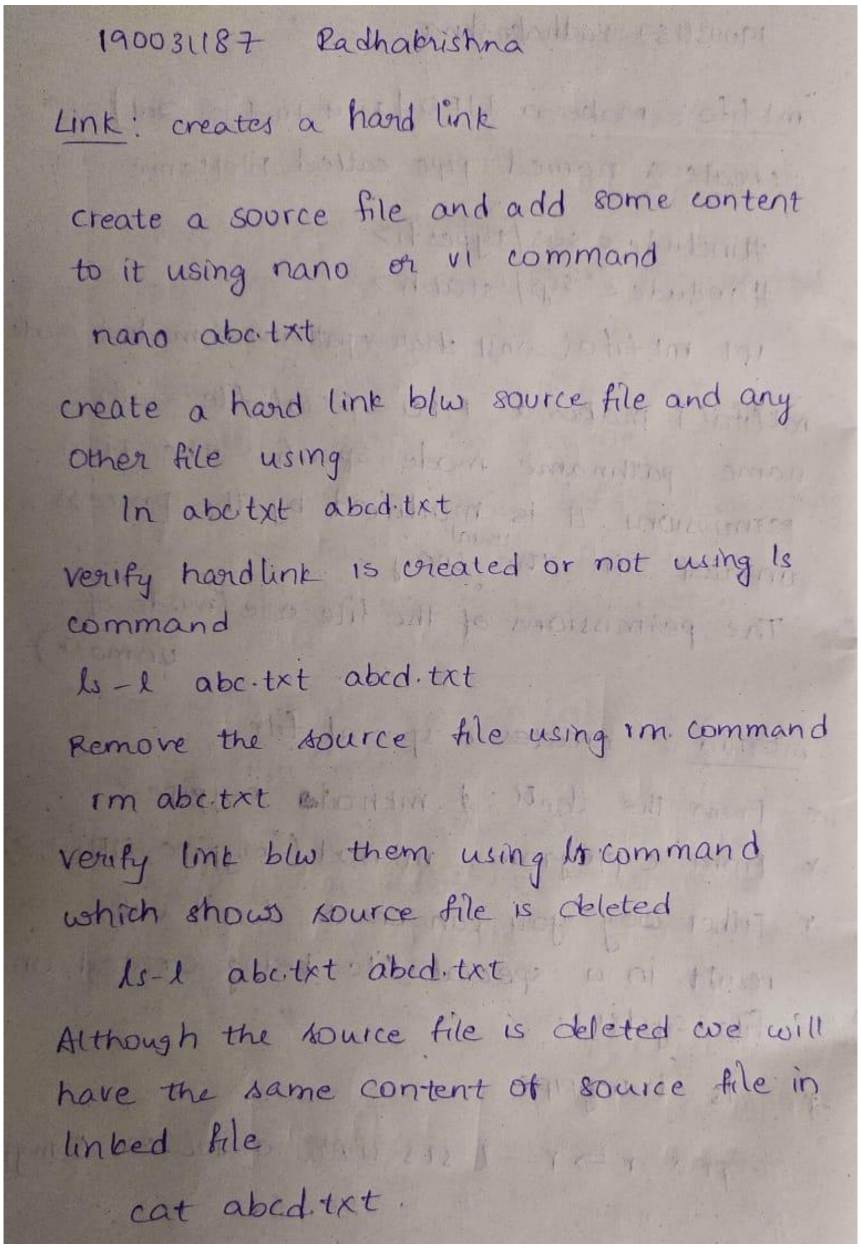
mknod: Creates a special file.



osd-190031187@team-osd:~

```
[osd-190031187@team-osd ~]$ mknod mypipe p
[osd-190031187@team-osd ~]$ ls -l mypipe
prw-rw-r--. 1 osd-190031187 osd-190031187 0 Sep  8 13:06 mypipe
[osd-190031187@team-osd ~]$ 
```

link: Creates a hard link.



190031187 Radhakrishna

Link: creates a hard link

Create a source file and add some content to it using nano or vi command

nano abc.txt

create a hard link b/w source file and any other file using

ln abc.txt abcd.txt

verify hardlink is created or not using ls command

ls -l abc.txt abcd.txt

Remove the source file using rm command

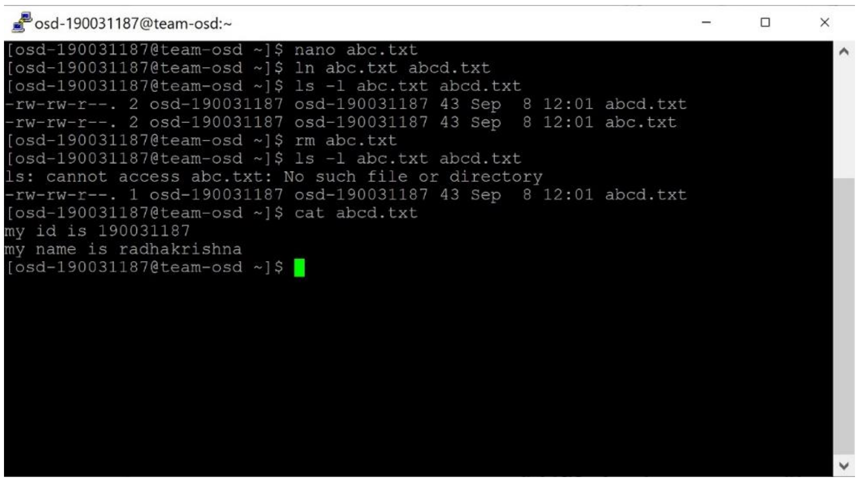
rm abc.txt

verify link b/w them using ls command which shows source file is deleted

ls-l abc.txt abcd.txt

Although the source file is deleted we will have the same content of source file in linked file

cat abcd.txt

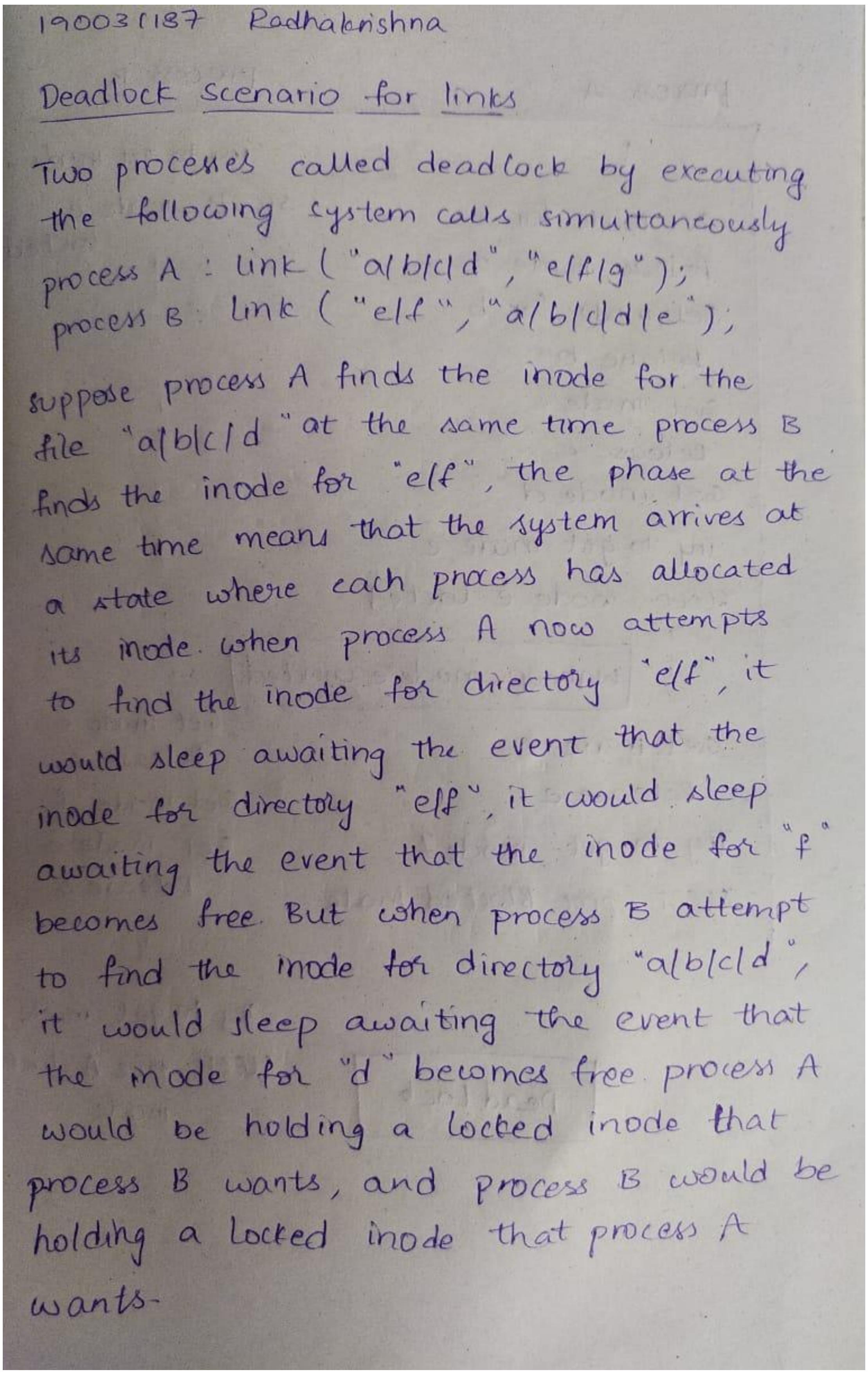


osd-190031187@team-osd:~

```
[osd-190031187@team-osd ~]$ nano abc.txt
[osd-190031187@team-osd ~]$ ln abc.txt abcd.txt
[osd-190031187@team-osd ~]$ ls -l abc.txt abcd.txt
-rw-rw-r--. 2 osd-190031187 osd-190031187 43 Sep  8 12:01 abcd.txt
-rw-rw-r--. 2 osd-190031187 osd-190031187 43 Sep  8 12:01 abc.txt
[osd-190031187@team-osd ~]$ rm abc.txt
[osd-190031187@team-osd ~]$ ls -l abc.txt abcd.txt
ls: cannot access abc.txt: No such file or directory
-rw-rw-r--. 1 osd-190031187 osd-190031187 43 Sep  8 12:01 abcd.txt
[osd-190031187@team-osd ~]$ cat abcd.txt
my id is 190031187
my name is radhakrishna
[osd-190031187@team-osd ~]$
```

190031187 **NERELLA VENKATA RADHAKRISHNA**

Deadlock scenario for link.



190031187 Radhakrishna

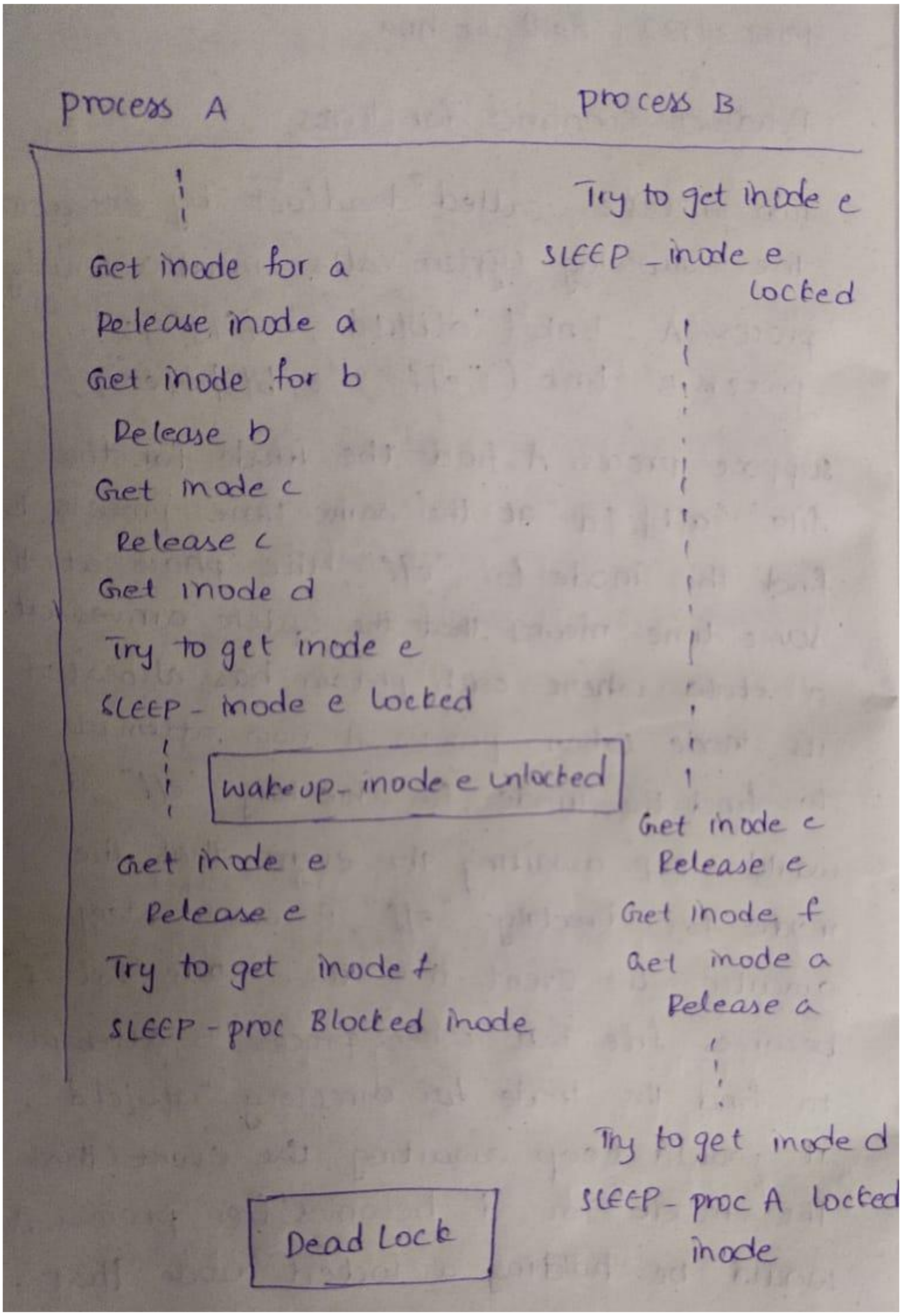
Deadlock scenario for links

Two processes called deadlock by executing the following system calls simultaneously

process A : link ("a/b/c/d", "e/f/g");

process B : link ("e/f", "a/b/c/d/e");

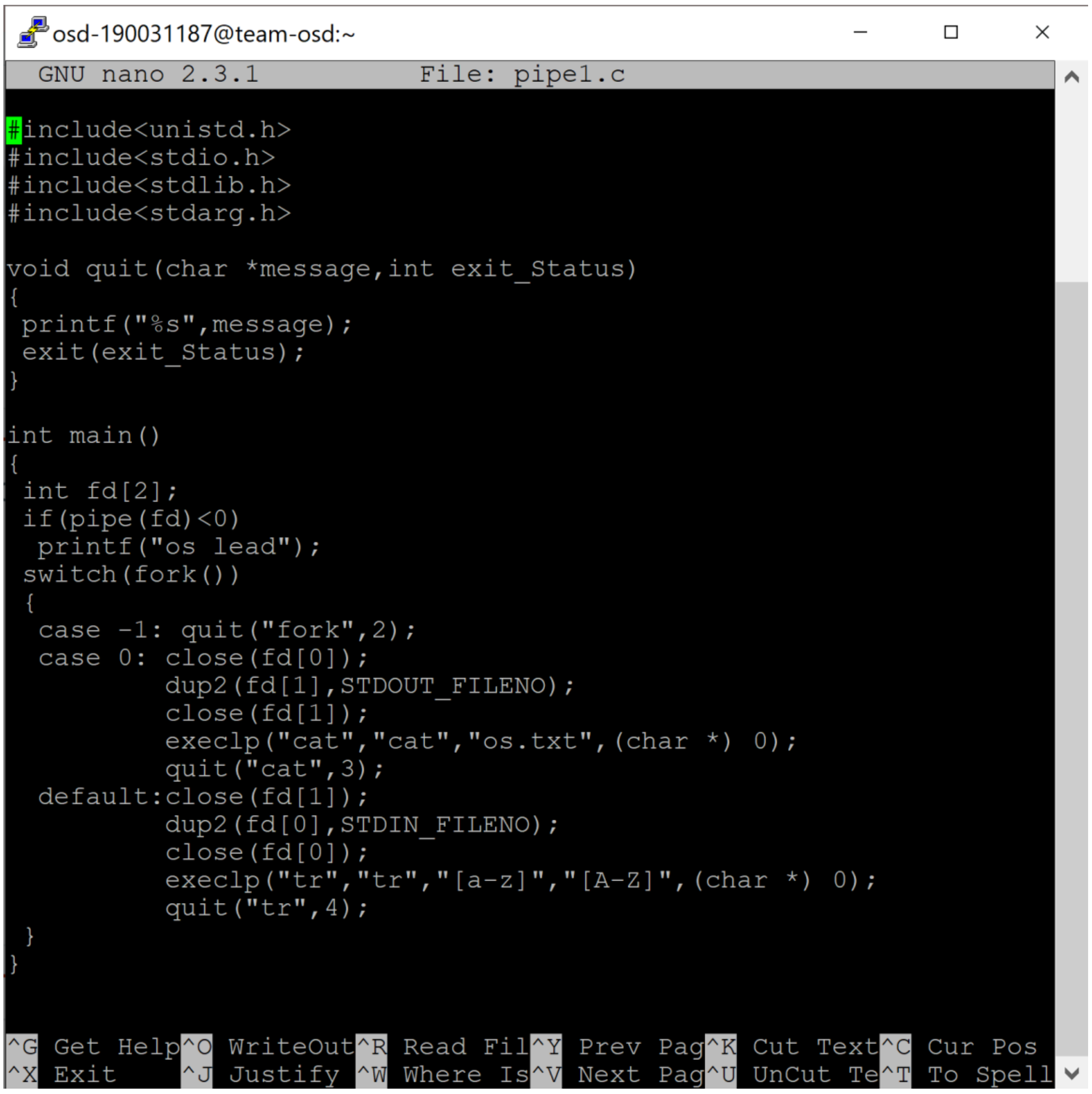
suppose process A finds the inode for the file "a/b/c/d" at the same time process B finds the inode for "e/f", the phase at the same time means that the system arrives at a state where each process has allocated its inode. when process A now attempts to find the inode for directory "e/f", it would sleep awaiting the event that the inode for directory "e/f", it would sleep awaiting the event that the inode for "f" becomes free. But when process B attempt to find the inode for directory "a/b/c/d", it would sleep awaiting the event that the inode for "d" becomes free process A would be holding a locked inode that process B wants, and process B would be holding a Locked inode that process A wants.



process A	process B
⋮	Try to get inode e
Get inode for a	SLEEP _ inode e locked
Release inode a	⋮
Get inode for b	⋮
Release b	⋮
Get inode c	⋮
Release c	⋮
Get inode d	⋮
Try to get inode e	⋮
SLEEP - inode e locked	⋮
⋮ $\boxed{\text{wakeup- inode e unlocked}}$	⋮
Get inode e	Get inode c
Release e	Release e
Try to get inode f	Get inode f
	Get inode a
SLEEP - proc Blocked inode	Release a
	⋮
	Try to get inode d
$\boxed{\text{Dead Lock}}$	SLEEP - proc A locked inode

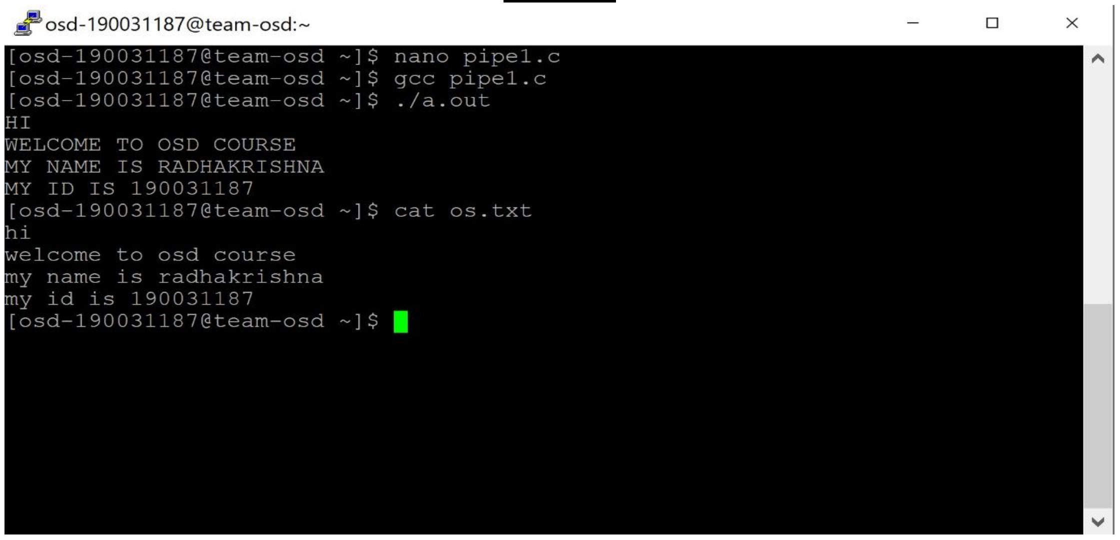
In-Lab:

1. **pipe.c**: Runs two programs in a pipeline Child runs cat, parent runs tr



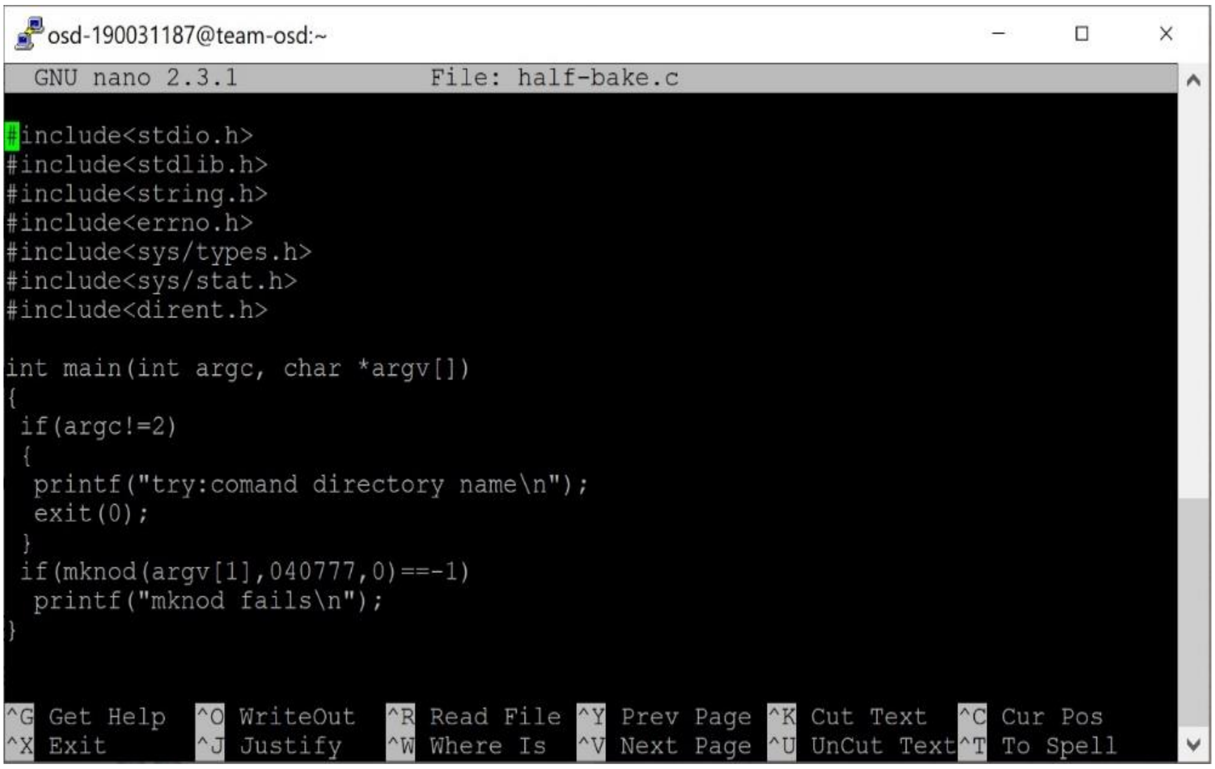
```
osd-190031187@team-osd:~
  GNU nano 2.3.1             File: pipe1.c

#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<stdarg.h>

void quit(char *message,int exit_Status)
{
 printf("%s",message);
 exit(exit_Status);
}

int main()
{
 int fd[2];
 if(pipe(fd)<0)
  printf("os lead");
 switch(fork())
 {
  case -1: quit("fork",2);
  case 0: close(fd[0]);
          dup2(fd[1],STDOUT_FILENO);
          close(fd[1]);
          execlp("cat","cat","os.txt",(char *) 0);
          quit("cat",3);
  default:close(fd[1]);
          dup2(fd[0],STDIN_FILENO);
          close(fd[0]);
          execlp("tr","tr","[a-z]","[A-Z]",(char *) 0);
          quit("tr",4);
 }
}

^G Get Help^O WriteOut^R Read Fil^Y Prev Pag^K Cut Text^C Cur Pos
^X Exit    ^J Justify ^W Where Is^V Next Pag^U UnCut Te^T To Spell
```

OUTPUT



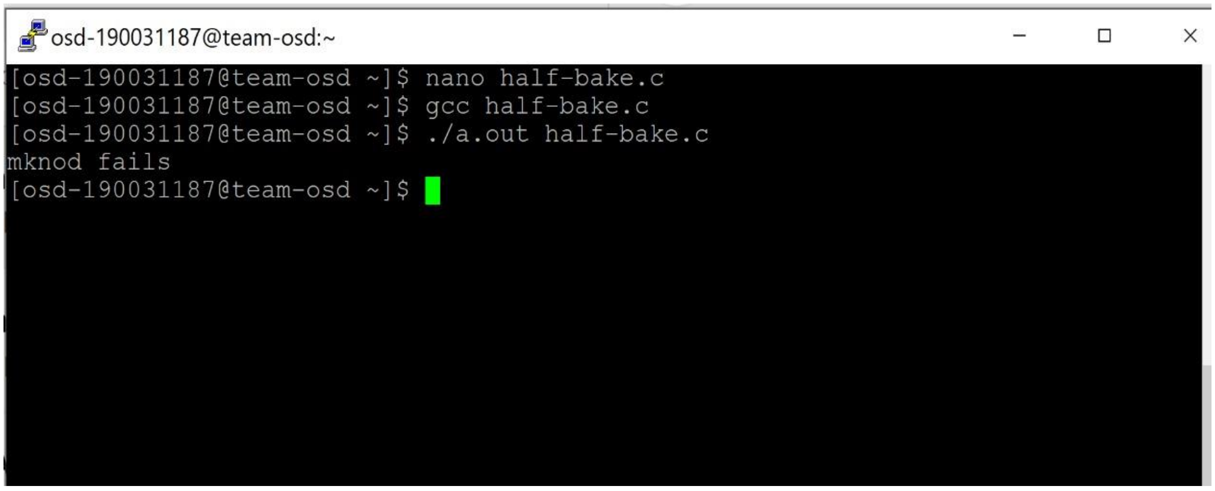
```
osd-190031187@team-osd:~
[osd-190031187@team-osd ~]$ nano pipe1.c
[osd-190031187@team-osd ~]$ gcc pipe1.c
[osd-190031187@team-osd ~]$ ./a.out
HI
WELCOME TO OSD COURSE
MY NAME IS RADHAKRISHNA
MY ID IS 190031187
[osd-190031187@team-osd ~]$ cat os.txt
hi
welcome to osd course
my name is radhakrishna
my id is 190031187
[osd-190031187@team-osd ~]$
```

2. A half-baked directory using mknod.



```
  GNU nano 2.3.1              File: half-bake.c

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<errno.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<dirent.h>

int main(int argc, char *argv[])
{
 if(argc!=2)
 {
  printf("try:comand directory name\n");
  exit(0);
 }
 if(mknod(argv[1],040777,0)==-1)
  printf("mknod fails\n");
}


^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

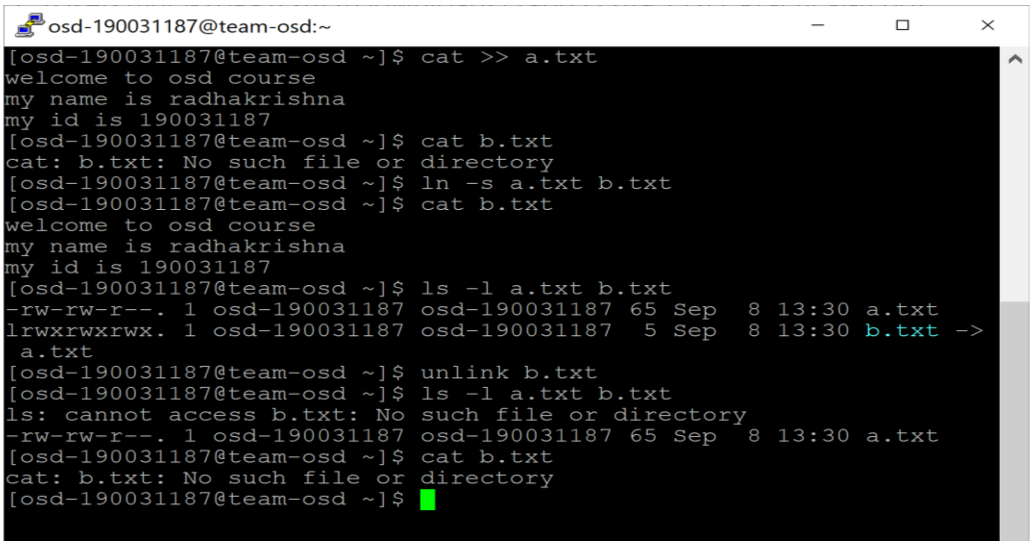
OUTPUT



```
[osd-190031187@team-osd ~]$ nano half-bake.c
[osd-190031187@team-osd ~]$ gcc half-bake.c
[osd-190031187@team-osd ~]$ ./a.out half-bake.c
mknod fails
[osd-190031187@team-osd ~]$ 
```

Post-Lab:

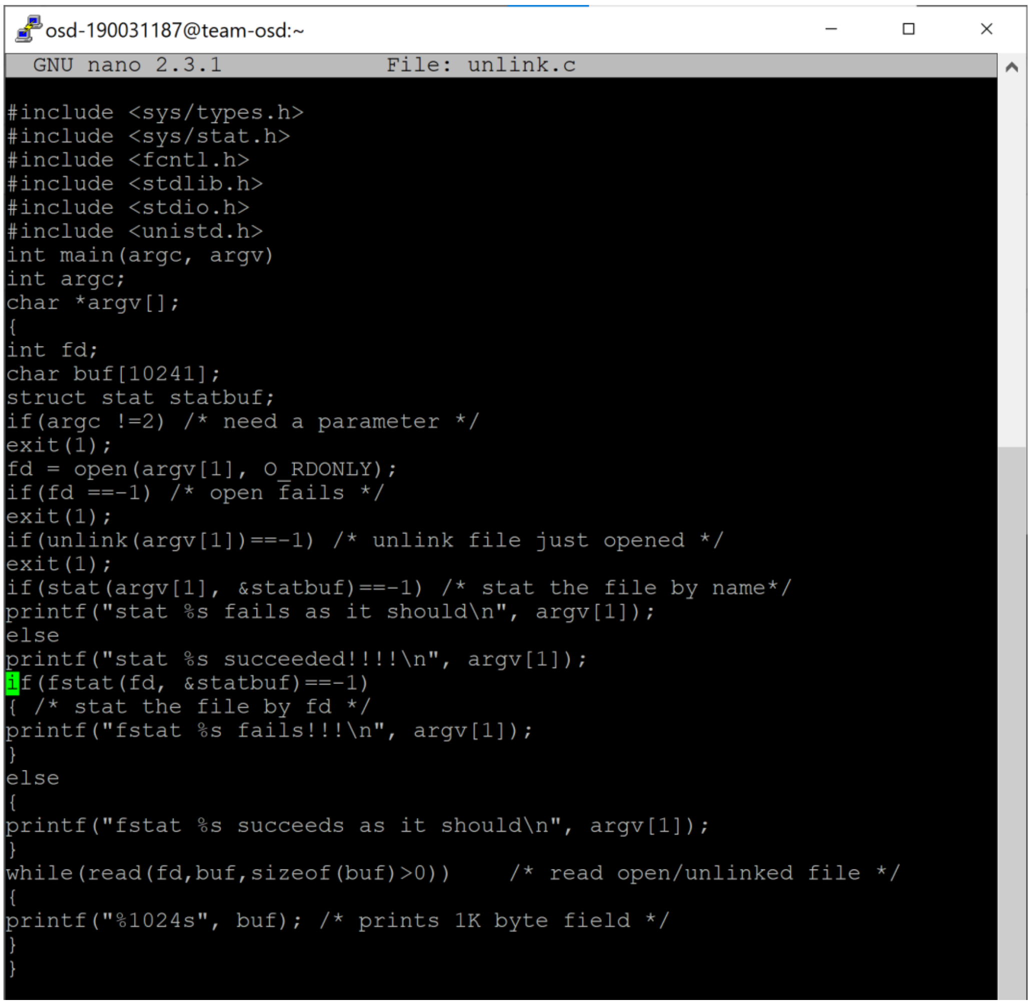
1. mylink.c -- create the filename "another.txt" and link it to the other file. Later delete it using unlink.



osd-190031187@team-osd:~

```
[osd-190031187@team-osd ~]$ cat >> a.txt
welcome to osd course
my name is radhakrishna
my id is 190031187
[osd-190031187@team-osd ~]$ cat b.txt
cat: b.txt: No such file or directory
[osd-190031187@team-osd ~]$ ln -s a.txt b.txt
[osd-190031187@team-osd ~]$ cat b.txt
welcome to osd course
my name is radhakrishna
my id is 190031187
[osd-190031187@team-osd ~]$ ls -l a.txt b.txt
-rw-rw-r--. 1 osd-190031187 osd-190031187 65 Sep  8 13:30 a.txt
lrwxrwxrwx. 1 osd-190031187 osd-190031187  5 Sep  8 13:30 b.txt ->
 a.txt
[osd-190031187@team-osd ~]$ unlink b.txt
[osd-190031187@team-osd ~]$ ls -l a.txt b.txt
ls: cannot access b.txt: No such file or directory
-rw-rw-r--. 1 osd-190031187 osd-190031187 65 Sep  8 13:30 a.txt
[osd-190031187@team-osd ~]$ cat b.txt
cat: b.txt: No such file or directory
[osd-190031187@team-osd ~]$
```

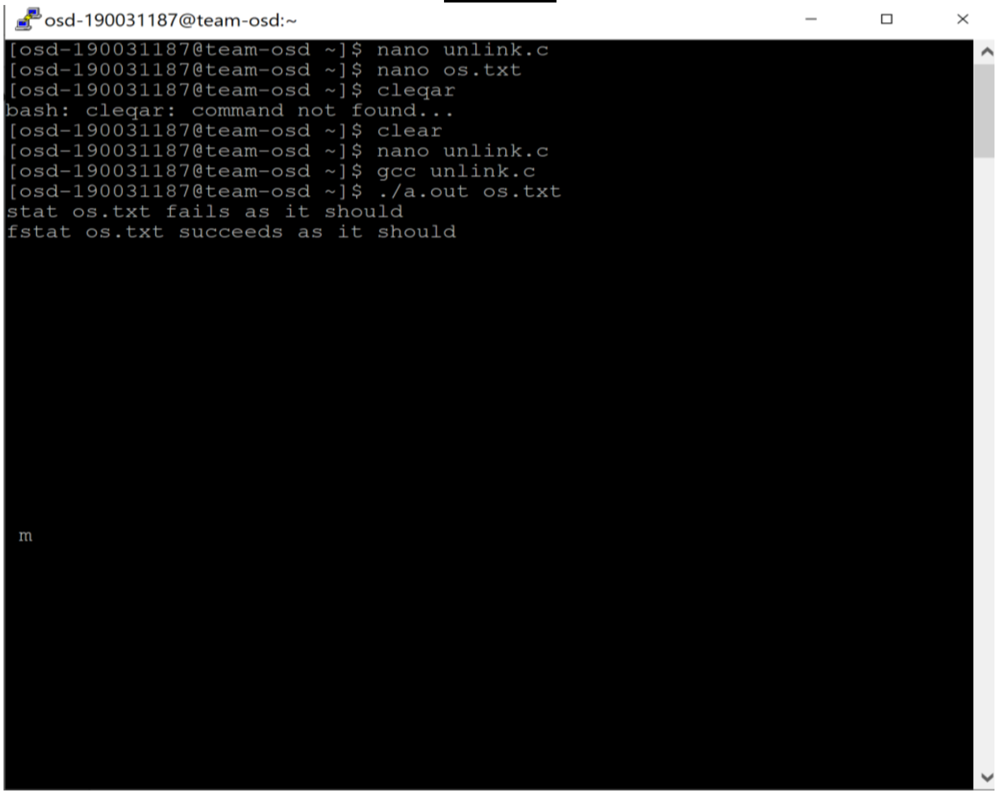
2. Unlinking an opened file



osd-190031187@team-osd:~

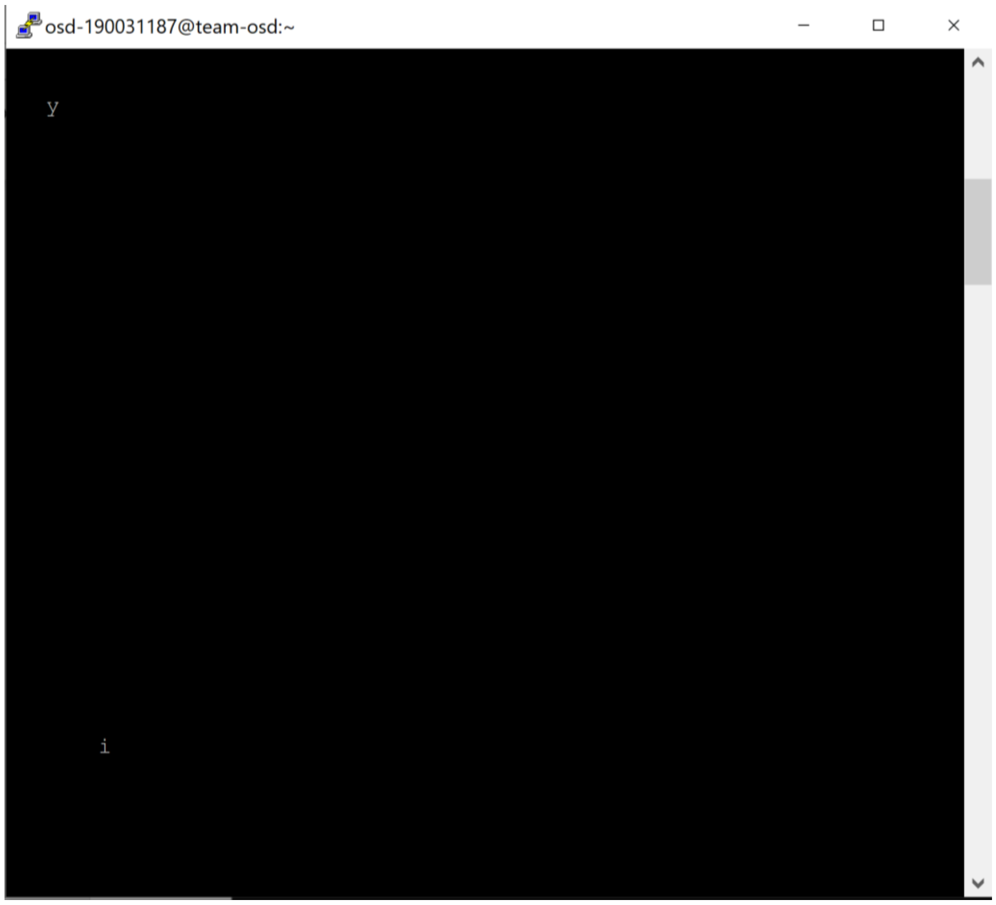
```
  GNU nano 2.3.1              File: unlink.c

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
int main(argc, argv)
int argc;
char *argv[];
{
int fd;
char buf[10241];
struct stat statbuf;
if(argc !=2) /* need a parameter */
exit(1);
fd = open(argv[1], O_RDONLY);
if(fd ==-1) /* open fails */
exit(1);
if(unlink(argv[1])==-1) /* unlink file just opened */
exit(1);
if(stat(argv[1], &statbuf)==-1) /* stat the file by name*/
printf("stat %s fails as it should\n", argv[1]);
else
printf("stat %s succeeded!!!!\n", argv[1]);
if(fstat(fd, &statbuf)==-1)
{ /* stat the file by fd */
printf("fstat %s fails!!!\n", argv[1]);
}
else
{
printf("fstat %s succeeds as it should\n", argv[1]);
}
while(read(fd,buf,sizeof(buf)>0))    /* read open/unlinked file */
{
printf("%1024s", buf); /* prints 1K byte field */
}
}
```

OUTPUT



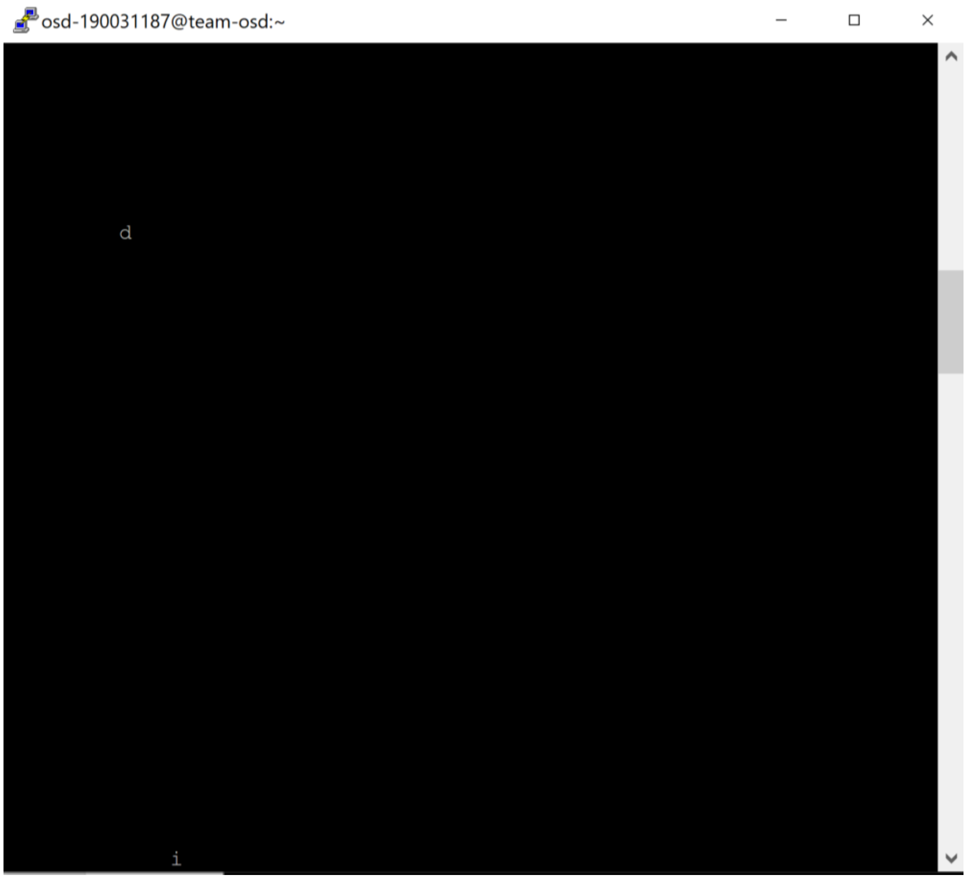
osd-190031187@team-osd:~

```
[osd-190031187@team-osd ~]$ nano unlink.c
[osd-190031187@team-osd ~]$ nano os.txt
[osd-190031187@team-osd ~]$ cleqar
bash: cleqar: command not found...
[osd-190031187@team-osd ~]$ clear
[osd-190031187@team-osd ~]$ nano unlink.c
[osd-190031187@team-osd ~]$ gcc unlink.c
[osd-190031187@team-osd ~]$ ./a.out os.txt
stat os.txt fails as it should
fstat os.txt succeeds as it should




 m
```



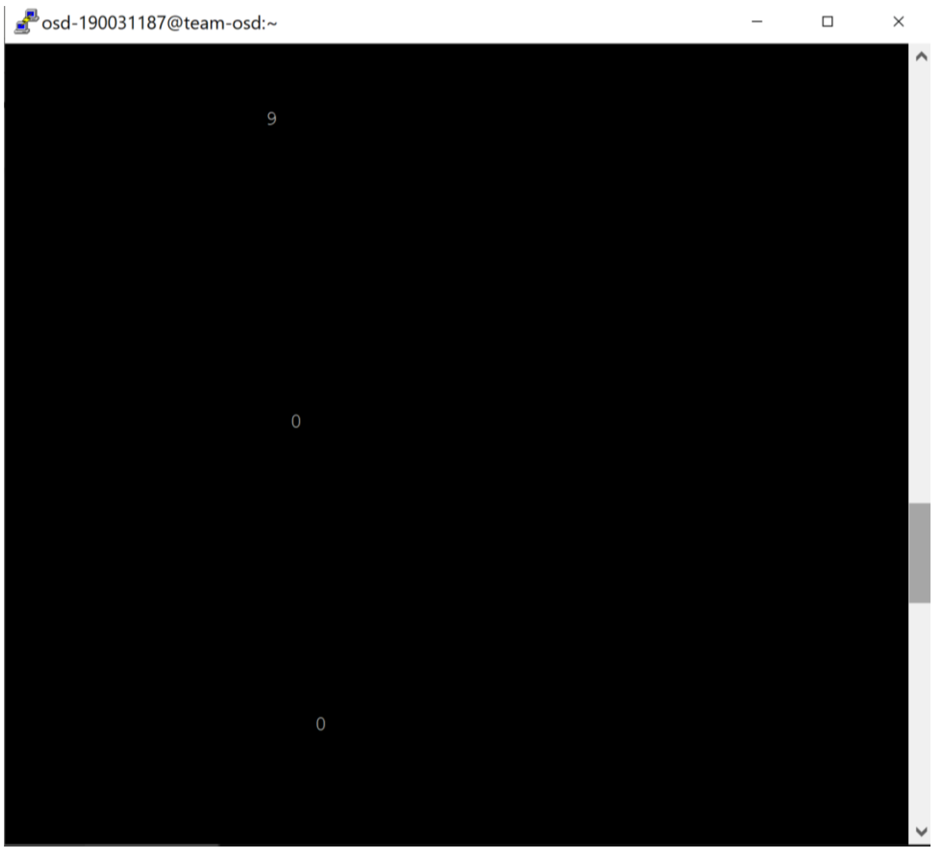
osd-190031187@team-osd:~

```

   y








       i
```

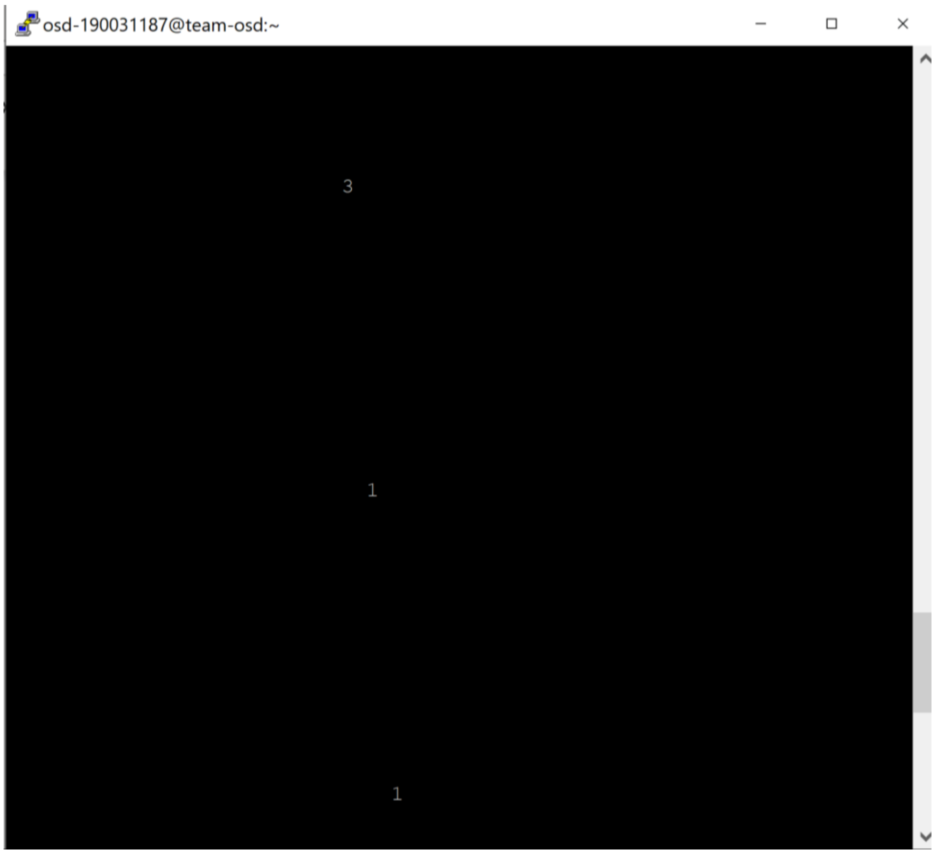


```
osd-190031187@team-osd:~

    d




                         i
```

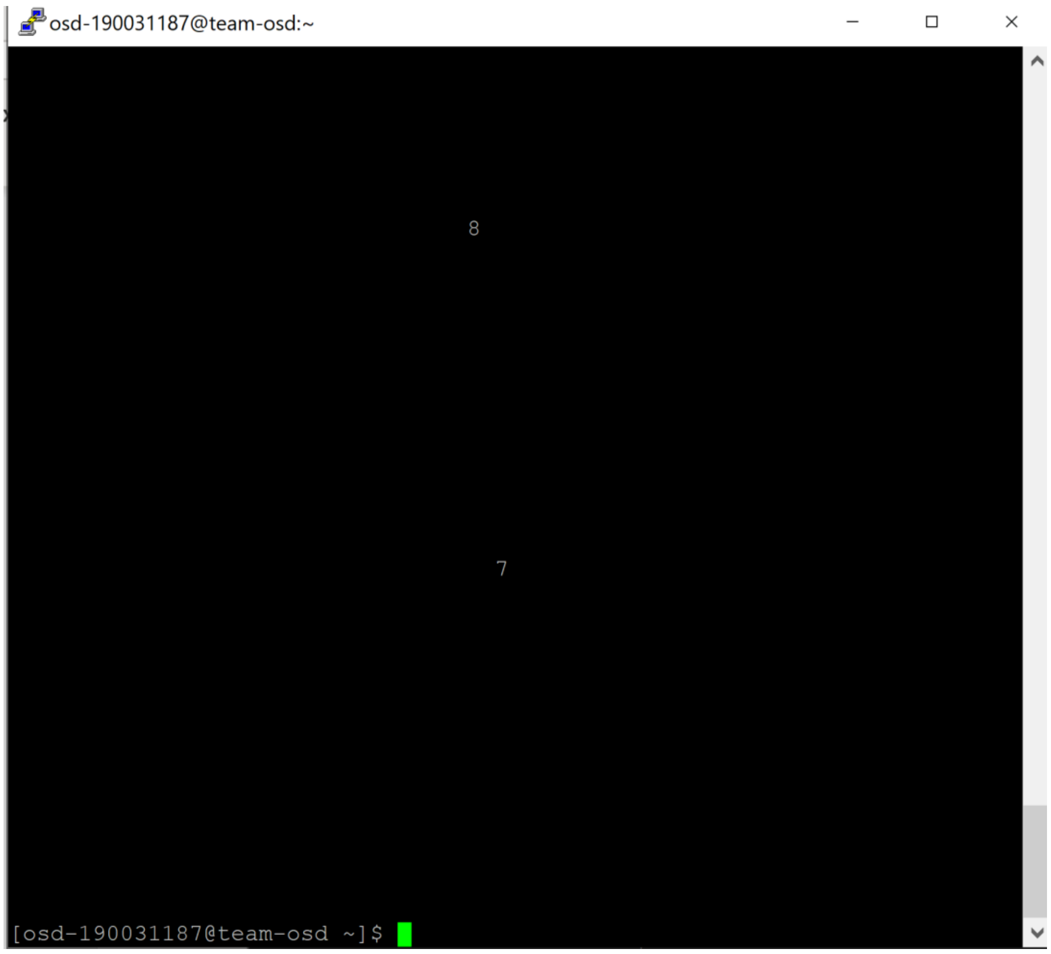
```
osd-190031187@team-osd:~

          s




                              1
```


osd-190031187@team-osd:~

```
          9



           0



            0
```


osd-190031187@team-osd:~

```
              3



               1



                1
```



```
                        8






                          7







[osd-190031187@team-osd ~]$
```